RED HAT ENTERPRISE LINUX 7: APPLICATION COMPATIBILITY GUIDE

June 2014

Version 7.0.100

www.redhat.com

Table of Contents

EXECUTIVE SUMMARY

This guide is intended to educate independent software vendors (ISVs) and customers on Red Hat's guidelines regarding support of third-party applications across multiple releases of the Red Hat Enterprise Linux platform. ISV and customer applications can reduce or avoid migration issues between major Red Hat® Enterprise Linux® versions by following these guidelines during application development.

This document describes the recommended uses of the system application programming (API) and binary interfaces (ABI) that are intended to provide compatibility across Red Hat Enterprise Linux releases. It outlines the tiered framework under which applications are considered compatible or not compatible.

INTRODUCTION

These guidelines seek to provide stability for applications when new releases are deployed and therefore focus primarily on forward-compatibility issues. Although backward compatibility is the goal, development of new capabilities sometimes makes that impractical. Hence, the guidelines and published policy may sometimes be over-ridden by the objective of providing our customers with the most competitive and capable systems possible; situational testing is always recommended.

If clarification of these compatibility guidelines is required, please see the Red Hat Enterprise Linux 7 Developer Guide[1] for additional details, or contact your Red Hat representative.

1 http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/7/html/Developer_Guide/index.html

TERMINOLOGY

The following are basic terms used in this document:

- **Binary compatibility**

  Binary compatibility means applications that are compiled on a combination of Red Hat Enterprise Linux and a particular hardware architecture will load and run similarly across different instances of the operating environment. Application binaries consist of executable files and Dynamic Shared Objects (DSO), and the level of compatibility is defined by a specific application binary interface (ABI).

- **Application programming interface (API)**

  An API is an interface implemented by a software program that enables it to interact with other software, including operating system components. The API is enforced at compile time and determines source compatibility, that is, whether application source code will compile similarly across different instances of the operating environment.

- **Application binary interface (ABI)**

  An ABI is a set of runtime conventions that interact with a compiled binary representation of a program. An ABI is a superset of an API and describes the low-level interface between the application and the operating environment. It covers details such as:

  - data type, size, and alignment
  - the calling standard, which defines how function arguments are passed and return values retrieved
  - the binary format of object files and program libraries
  - function and / or data symbol names and their versions

  Tools such as the compilers, linkers, runtime libraries, and the operating system itself need to work with the ABI. An ABI operates at runtime.

- **ABI conformance**

  A compiler conforms to an ABI if it generates code that follows all of the specifications enumerated by that ABI. A library conforms to an ABI if it is implemented according to that ABI. An application conforms to an ABI if it is built using tools that conform to that ABI and does not contain source code that changes behavior specified by the ABI or that otherwise bypasses the ABI.

- **Core persistent system infrastructure**

  The core persistent system infrastructure refers to interfaces and externally available data structures that represent system state or provide a means of communicating with the system (for instance, system calls and header files).

- **Compatibility in a virtualized environment**

  Virtual environments emulate bare-metal environments such that unprivileged applications that run on bare-metal environments will run, unmodified, in corresponding virtual environments. Virtual

environments present simplified abstracted views of physical resources, so some differences may exist.

- **Packages in Supplementary and Optional repositories**

  Packages available in the Red Hat Network (RHN) Supplementary and Optional channels are not part of the product but are made available for convenience or to satisfy build dependencies.

- **Major and minor releases**

  A Red Hat major release represents a significant step in the development of a product (wholesale changes are usually reserved for major releases), and is typically designated by a single numeral (e.g., Red Hat Enterprise Linux 7). Minor releases appear more frequently, within the scope of a major release, and generally represent smaller, incremental developmental steps.

- **Red Hat Enterprise Linux Extended Update Support**

  Red Hat Enterprise Linux Extended Update Support (EUS) is an optional support offering that allows a customer to standardize on a specific minor release for an extended period of time. Each Red Hat Enterprise Linux 7 EUS stream is available for 24 months from the availability of the minor release.

- **Red Hat Enterprise Linux Advanced Update Support**

  Red Hat Enterprise Linux Advanced Update Support (AUS) is an optional support offering the allows a customer to stay on a specific minor release for up to 24 months and receive only critical errata.

- **Red Hat Developer Toolset (DTS)**

  Red Hat Developer Toolset (DTS) is an offering that provides the latest stable versions of compilers, debuggers, and select open source development tools. This optional offering has an independent life cycle and is not covered by this Application Compatibility document.

- **Red Hat Software Collections (RHSCL)**

  Red Hat Software Collections delivers the latest, stable versions of some of the most popular web development languages and open source databases for use with Red Hat Enterprise Linux. This optional offering has an independent life cycle and is not covered by the guidelines described in this Application Compatibility document.

- **RHEL Extras**

  The contents of the Extras channel are exempt from Red Hat's policy that commits to binary compatibility. See https://access.redhat.com/site/solutions/5154 for details about this binary compatibility policy.

- **SystemTap Static Probes**

  SystemTap static probes are part of the SystemTap profiling and tracing framework. The probes are integrated into key system libraries to support profiling and debugging of applications and libraries. No assurances are made at this time that integrated SystemTap static probes will continue to have the same probe name, probe location, or interpretation or number of arguments.

SCOPE OF COMPATIBILITY

Packages in Red Hat Enterprise Linux are classified under one of the following four compatibility levels:

- Compatibility level 1: APIs and ABIs are stable across three major releases; the release that introduces a new or revised API or ABI, and the two following major releases (*n*, *n*+1, *n*+2). In the case of this document, release *n* starts with Red Hat Enterprise Linux 7.

  If a change to a library during its life cycle causes an incompatibility with existing compiled code, a separate version of the library will be provided with the older ABI to run the application without modifications.
- Compatibility level 2: APIs and ABIs are stable within one major release. In the case of this document, the major release is Red Hat Enterprise Linux 7.
- Compatibility level 3: *Reserved for future use*.
- Compatibility level 4: No compatibility is provided. This includes all libraries that the listed package supplies.

For compatibility levels for specific Red Hat Enterprise Linux software packages, please see Appendix A.

Compatibility levels for bare-metal configurations apply to virtualized configurations except for any features that directly interact with hardware. Those features directly related to hardware have no API or ABI compatibility level. For example, applications that rely on Graphics Processing Units (GPU) features cannot expect binary compatibility.

Red Hat strongly recommends that application developers validate that any behavior they depend on is explicitly defined in formal API documentation to prevent introducing dependencies on unspecified implementation-specific semantics or introducing dependencies on bugs in a particular implementation of an API. For example, new releases of the GNU C library may not be compatible with older releases if an application used an undocumented API or one with undefined behavior.

GUIDELINES FOR PRESERVING BINARY COMPATIBILITY

Red Hat recommends that application developers adopt the following principles in order to improve binary compatibility:

1. Build applications using the published interfaces of a library. Non-published (a.k.a. internal) interfaces are subject to change at any time, which can cause instability in the dependent application if improperly linked. Application developers should validate that any behavior they depend on is described in the published API documentation to prevent introducing dependencies on unspecified implementation-specific semantics or introducing dependencies on bugs in a particular implementation of an API. For example, new releases of the GNU C library are not guaranteed to be compatible with older releases if the old behavior was not consistent with a published specification.
2. Avoid static linking of libraries (C/C++). Static linking causes the executables to have their own version of the library. This increases the chance of an application not running predictably on a later

version of the operating system as these library dependencies might have changed along the way. Linking applications dynamically is strongly recommended in order to avoid this problem.

3. Limit linking applications to the core libraries at compatibility level 1. The core libraries (see Appendix A) are intended to preserve binary compatibility across three consecutive major releases.
4. Provide compatibility libraries for applications that have been built with libraries that are not at the desired compatibility level, provided the bundled libraries themselves only use the interfaces provided by the core libraries.
5. Package applications using the RPM mechanism. RPM provides a software-packaging mechanism that includes detailed specification of application dependencies. When creating RPMs, the following should be kept in mind:
   (a) Avoid using RPM triggers whenever possible.
   (b) Explicitly state all required runtime and build dependencies using the appropriate RPM syntax.
   (c) Do not modify, replace, or recompile files managed by Red Hat-provided RPM packages. Doing so may lead to unpredictable behavior.
   (d) When considering dependencies, do not assume that all possible packages will be installed on every Red Hat Enterprise Linux system. The default installed packages may change between major releases, between product variants of the same version, and on a customer's system. For instance, the Workstation product will have a different installed package set than the Server product. Rarely, and due to extenuating circumstances, packages might be removed between minor releases but Red Hat will provide notification if this occurs.
6. Follow the Filesystem Hierarchy Standard (FHS) version 2.3 when installing programs. Third-party software should be installed to the '/opt' subdirectory. More information on the FHS is available at: http://www.pathname.com/fhs/.
7. Applications should be built against libraries that correspond to the native hardware environment rather than a compatibility layer on top of the hardware. This is because the compatibility userspace contains a subset of system libraries as compared to the native userspace.
8. Do not design applications that rely on configuration files used by system packages. These files can change between major releases unless the upstream community is explicitly committed to preserving them.
9. If you require functionality from a package that is currently at a lower compatibility level than you wish, please contact Red Hat for review.

Note: During the life cycle of a major release, Red Hat makes commercially reasonable efforts to maintain binary compatibility for the core runtime environment across all minor releases and errata advisories. If necessary, Red Hat may make exceptions to this compatibility goal for critical impact security or other significant issues. Furthermore, as described above and in Appendix A, major releases of Red Hat Enterprise Linux contain a limited set of backward-compatible libraries included in previous major releases to allow for the easy migration of applications. Typically, Red Hat applies changes in such a way as to minimize the amount of change and to maintain binary compatibility. Exceptions may apply for controlled package re-bases under certain circumstances.

APPENDIX A: COMPATIBILITY LEVELS FOR SPECIFIC PACKAGES AND LIBRARIES

Compatibility level 1: APIs and ABIs are stable across three major releases (starting with RHEL 7)

alsa-lib	krb5-libs	libtbbmalloc	libxslt
elfutils-libelf	libgcc	libtbbmalloc-proxy	mesa-libGL
glibc	libgfortran	libtopology	mesa-libGLU
glibc-utils	libgomp	libusb	motif
gtk2	libstdc++	libvirt-client	pam
hesiod	libtbb	libxml2	SDL

Compatibility level 2: APIs and ABIs are stable within one major release (RHEL 7)

atk	kdebase-workspace-libs	libXft	tcl
audit-libs	kdegraphics	libXi	tcp_wrappers-libs
audit-libs-python	kdegraphics-libs	libXinerama	tk
boost-date-time	kdelibs	libXmu	unique
boost-filesystem	kdemultimedia	libXpm	xz-libs
boost-graph	kdemultimedia-libs	libXrandr	zlib
boost-iostreams	kdenetwork	libXrender	
boost-math	kdenetwork-libs	libXt	
boost-program-options	kdepim	libXtst	
boost-python	kdepim-libs	mysql-libs	
boost-regex	kdepimlibs	ncurses-libs	
boost-serialization	kdepimlibs-akonadi	net-snmp-libs	
boost-signals	kdesdk	net-snmp-perl	
boost-system	kdesdk-devel	net-snmp-python	
boost-test	kdesdk-libs	nss	
boost-thread	kdesdk-utils	nss-sysinit	
boost-wave	kdm	numactl	
bzip2-libs	libacl	p11kit	
c-ares	libaio	pango	
cairo	libatomic	papi[4]	
clutter	libattr	pcre	
corosynclib	libblkid	perl	
cups-libs	libcanberra	perl-Digest-SHA	

cyrus-sasl-gssapi	libcanberra-gtk2	perl-libs
cyrus-sasl-lib	libcanberra-gtk3	perl-Time-Piece
cyrus-sasl-md5	libcap-ng	phonon-backend-gstreamer
db4	libcurl	polkit
db4-cxx	libgudev1	popt
dbus-glib	libhugetlbfs	postgresql-libs
dbus-libs	libICE	pulseaudio-libs
ding-libs	libjpeg	pulseaudio-libs-glib2
elfutils-libs	libnotify	PyQt4
expat	libpfm[2]	python
fuse-libs	libpng	python-libs
GConf2	librsvg2	qt
gdk-pixbuf2	libSM	qt-mysql
glib2	libsmbclient	qt-odbc
gmp	libtalloc	qt-postgresql
gnome-keyring	libtdb	qt-x11
gnome-keyring-pam	libtevent	qt3
gnutls	libtiff	qt3-MySQL
gtk3	libudev	qt3-ODNC
gstreamer1	libusb	qt3-PostgreSQL
gstreamer-plugins-base	libuuid	readline
httpd	libverto	realmd
java-1.7.0-openjdk[3]	libwbclient	ruby
kdebase	libX11	scl-utils
kdebase-libs	libXau	sqlite
kdebase-workspace	libXaw	startup-notification
kdebase-workspace-akonadi	libXext	systemtap[4]

Compatibility level 3: Reserved for Future Use

(*this section intentionally left blank*)

2 Non-regression on same hardware only; Red Hat is not guaranteeing future hardware support.

3 Red Hat will maintain compatibility for OpenJDK based on the Java SE 7 specification guidelines. Red Hat will defer to the Java SE specification for compatibility between the respective system/default Java runtime between major versions of RHEL. In most cases, compatibility between major versions of RHEL can be maintained by using the same version of OpenJDK. If that is not an option due to customer preference or Java lifecycle limitations, the Java SE specification should be used as guidelines to determine if applications can be run unmodified between major versions of RHEL.

4 Source-level script compatibility

Compability level 4: Stability of APIs or ABIs subject to change at Red Hat's discretion

adcli	libcom_err	postgresql-plperl	valgrind
binutils (libraries only)	libdrm	postgresql-plpython	xfsprogs
dyninst	libldb	postgresql-pltcl	
e2fsprogs-libs	libitm	postgresql-server	
glade3-libgladeui	libss	postgresql-test	
gnome-desktop	libtsan	pulseaudio-libs-zeroconf	
gstreamer	mesa-dri-drivers	pulseaudio-module-bluetooth	
gstreamer-devel	mysql-server	pulseaudio-module-gconf	
gstreamer-plugins-base	mariadb-server	pulseaudio-module-x11	
gvfs	openldap	pulseaudio-utils	
junit	openslp-server	samba-libs	
libasan	postgresql-contrib	seekwatcher	
libcgroup	postgresql-docs	tkinter	